Vorlesung Graphenzeichnen:

Order in the Underground *or* How to Automate the Drawing of Metro Maps

Martin Nöllenburg

Lehrstuhl für Algorithmik I

26.06.2008

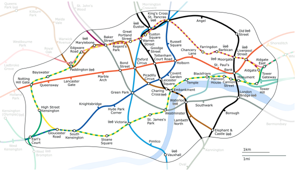
Outline

- Modeling the Metro Map Problem
  - What is a Metro Map?
  - Hard and Soft Constraints
- NP-Hardness: Bad News—Nice Proof
  - Rectilinear vs. Octilinear Drawing
  - Reduction from PLANAR 3-SAT
- MIP Formulation & Experiments
  - Mixed-Integer Programming Formulation
  - Experiments
  - Labeling

Outline

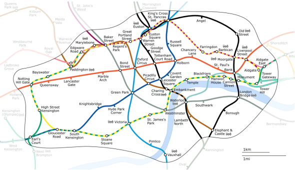
- Modeling the Metro Map Problem
  - What is a Metro Map?
  - Hard and Soft Constraints
- NP-Hardness: Bad News—Nice Proof
  - Rectilinear vs. Octilinear Drawing
  - Reduction from PLANAR 3-SAT
- MIP Formulation & Experiments
  - Mixed-Integer Programming Formulation
  - Experiments
  - Labeling

What is a Metro Map?



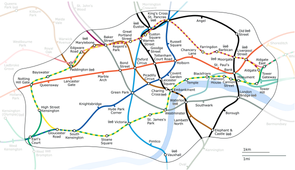
- schematic diagram for public transport

What is a Metro Map?



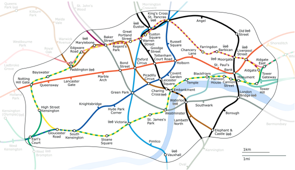
- schematic diagram for public transport
- visualizes lines and stations

What is a Metro Map?



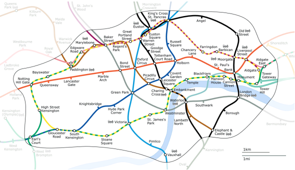
- schematic diagram for public transport
- visualizes lines and stations
- goal: ease navigation for passengers
  - “How do I get from A to B?”
  - “Where to get off and change trains?”

What is a Metro Map?



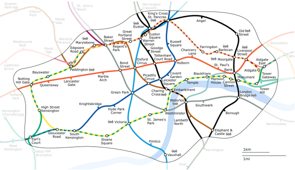
- schematic diagram for public transport
- visualizes lines and stations
- goal: ease navigation for passengers
  - “How do I get from A to B?”
  - “Where to get off and change trains?”
- distorts geometry and scale

What is a Metro Map?



- schematic diagram for public transport
- visualizes lines and stations
- goal: ease navigation for passengers
  - “How do I get from A to B?”
  - “Where to get off and change trains?”
- distorts geometry and scale
- improves readability

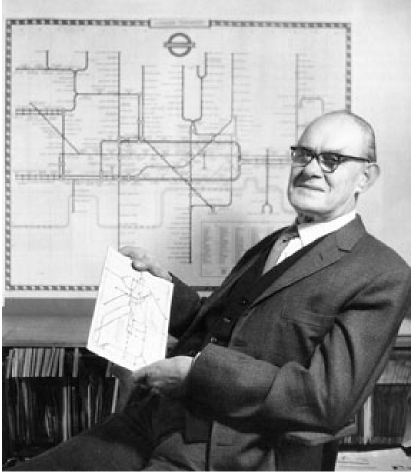
What is a Metro Map?



- schematic diagram for public transport
- visualizes lines and stations
- goal: ease navigation for passengers
    - “How do I get from A to B?”
    - “Where to get off and change trains?”
- distorts geometry and scale
- improves readability
- compromise between
  schematic road map ↔ abstract graph

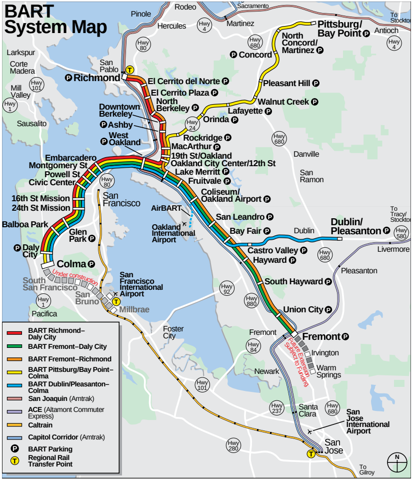
Why Automate Drawing Metro Maps?

- current maps designed manually



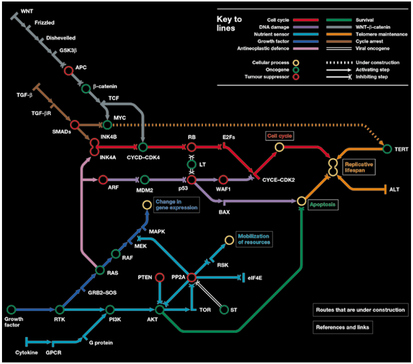
Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps



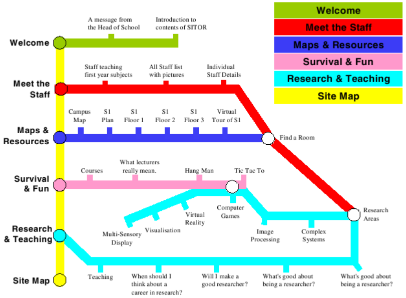
Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
  - metabolic pathways



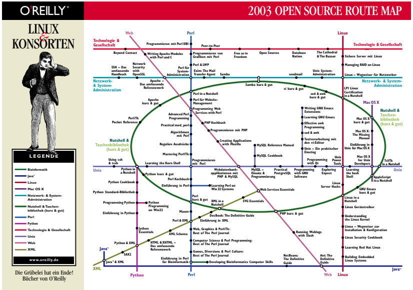
Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
  - metabolic pathways
  - web page maps



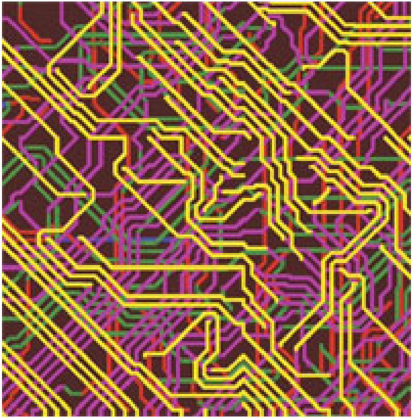
Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
  - metabolic pathways
  - web page maps
  - product lines



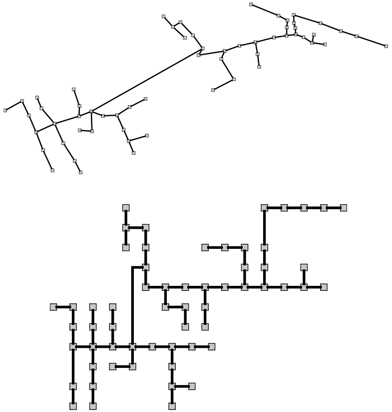
Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
  - metabolic pathways
  - web page maps
  - product lines
- VLSI: X-architecture



Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
  - metabolic pathways
  - web page maps
  - product lines
- VLSI: X-architecture
- drawing sketches
  [Brandes et al. '03]



More Formally

The Metro Map Problem

Given: planar embedded graph $G = (V, E)$, $V \subset \mathbb{R}^2$,
line cover $\mathcal{L}$ of paths or cycles in G (the metro lines),

Goal: draw G and $\mathcal{L}$ nicely.

More Formally

The Metro Map Problem

Given: planar embedded graph $G = (V, E)$, $V \subset \mathbb{R}^2$,
line cover $\mathcal{L}$ of paths or cycles in G (the metro lines),

Goal: draw G and $\mathcal{L}$ nicely.

- What is a nice drawing?

More Formally

The Metro Map Problem

Given: planar embedded graph $G = (V, E)$, $V \subset \mathbb{R}^2$,
line cover $\mathcal{L}$ of paths or cycles in G (the metro lines),

Goal: draw G and $\mathcal{L}$ nicely.

- What is a nice drawing?
- Look at real-world metro maps drawn by graphic designers and model their design principles as

More Formally

The Metro Map Problem

Given: planar embedded graph $G = (V, E)$, $V \subset \mathbb{R}^2$,
line cover $\mathcal{L}$ of paths or cycles in G (the metro lines),

Goal: draw G and $\mathcal{L}$ nicely.

- What is a nice drawing?
- Look at real-world metro maps drawn by graphic designers and model their design principles as
  - *hard* constraints – must be fulfilled,

More Formally

The Metro Map Problem

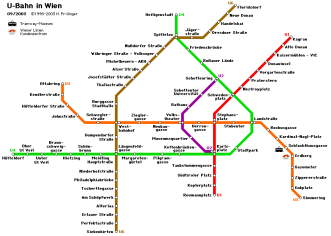
Given: planar embedded graph $G = (V, E)$, $V \subset \mathbb{R}^2$,
line cover $\mathcal{L}$ of paths or cycles in G (the metro lines),

Goal: draw G and $\mathcal{L}$ nicely.

- What is a nice drawing?
- Look at real-world metro maps drawn by graphic designers and model their design principles as
  - *hard* constraints – must be fulfilled,
  - *soft* constraints – should hold as tightly as possible.

Hard Constraints

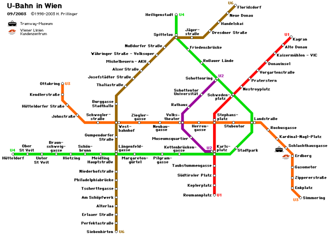
(H1) preserve embedding of G



Hard Constraints

(H1) preserve embedding of G

(H2) draw all edges as octilinear line segments,
i.e. horizontal, vertical or diagonal (45 degrees)



Hard Constraints

(H1) preserve embedding of G

(H2) draw all edges as octilinear line segments, i.e. horizontal, vertical or diagonal (45 degrees)

(H3) draw each edge e with length $\geq \ell_e$

Hard Constraints

(H1) preserve embedding of G

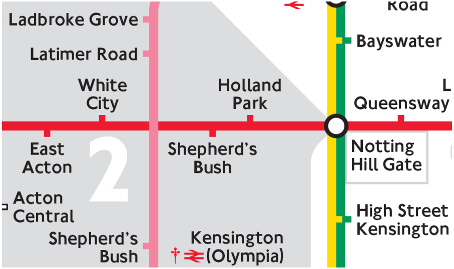
(H2) draw all edges as octilinear line segments, i.e. horizontal, vertical or diagonal (45 degrees)

(H3) draw each edge e with length $\geq \ell_e$

(H4) keep edges d_{min} away from non-incident edges

Soft Constraints

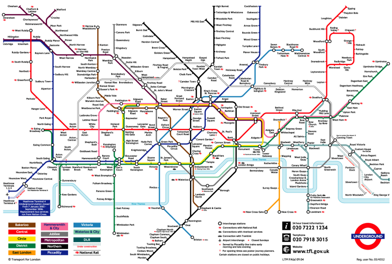
(S1) draw metro lines with few bends



Soft Constraints

(S1) draw metro lines with few bends

(S2) keep total edge length small

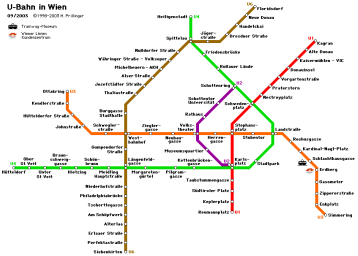


Soft Constraints

(S1) draw metro lines with few bends

(S2) keep total edge length small

(S3) draw each octilinear edge similar to its geographical orientation: keep its relative position



Outline

- Modeling the Metro Map Problem
  - What is a Metro Map?
  - Hard and Soft Constraints
- NP-Hardness: Bad News—Nice Proof
  - Rectilinear vs. Octilinear Drawing
  - Reduction from PLANAR 3-SAT
- MIP Formulation & Experiments
  - Mixed-Integer Programming Formulation
  - Experiments
  - Labeling

Another Problem

RECTILINEARGRAPHDRAWING Decision Problem

Given a planar embedded graph G with max degree 4.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is rectilinear?

Another Problem

RECTILINEARGRAPHDRAWING Decision Problem

Given a planar embedded graph G with max degree 4.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is rectilinear?

Theorem (Tamassia SIAMJComp'87)

RECTILINEARGRAPHDRAWING *can be solved efficiently.*

Another Problem

RECTILINEARGRAPHDRAWING Decision Problem

Given a planar embedded graph G with max degree 4.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is rectilinear?

Theorem (Tamassia SIAMJComp'87)

RECTILINEARGRAPHDRAWING *can be solved efficiently.*

Proof.

By reduction to a flow problem. □

Another Problem

RECTILINEARGRAPHDRAWING Decision Problem

Given a planar embedded graph G with max degree 4.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is rectilinear?

Theorem (Tamassia SIAMJComp'87)

RECTILINEARGRAPHDRAWING *can be solved efficiently.*

Proof.

By reduction to a flow problem. □

Our Problem

METROMAPLAYOUT Decision Problem

Given a planar embedded graph G with max degree 8.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is octilinear?

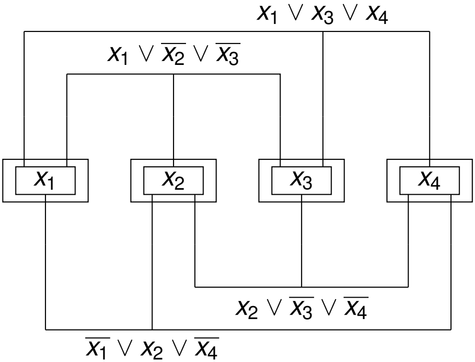
Theorem (Nöllenburg MSc'05)

METROMAPLAYOUT *is NP-hard.*

Proof.

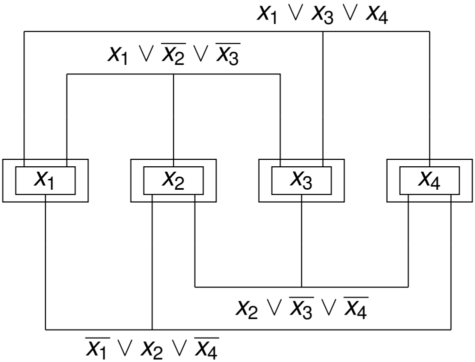
By Reduction from PLANAR 3-SAT to METROMAPLAYOUT. □

Outline of the Reduction



Input: planar 3-SAT formula $\varphi =$
$(x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \dots$

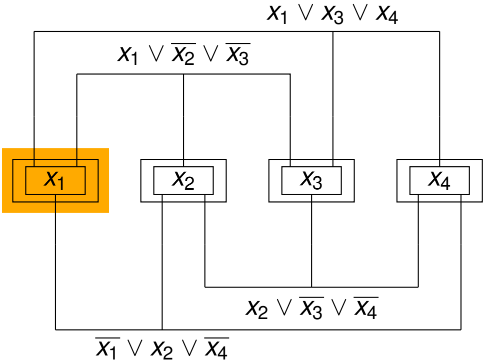
Outline of the Reduction



Input: planar 3-SAT formula $\varphi = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \dots$

Goal: planar embedded graph G_φ with: G_φ has a metro map drawing $\Leftrightarrow$ φ satisfiable.

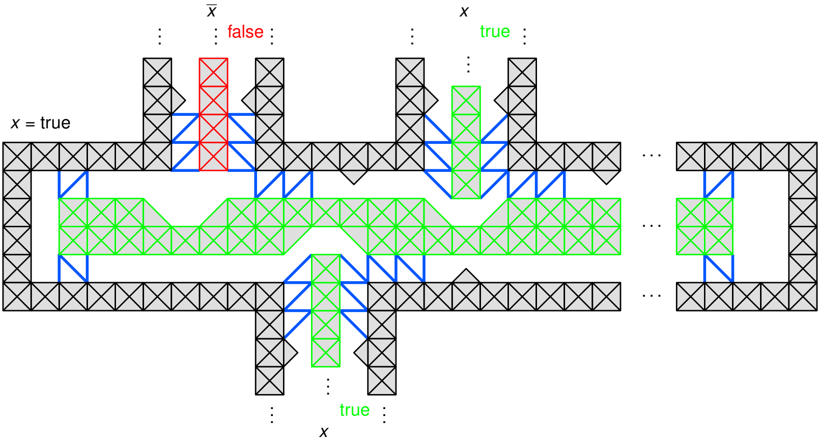
Outline of the Reduction



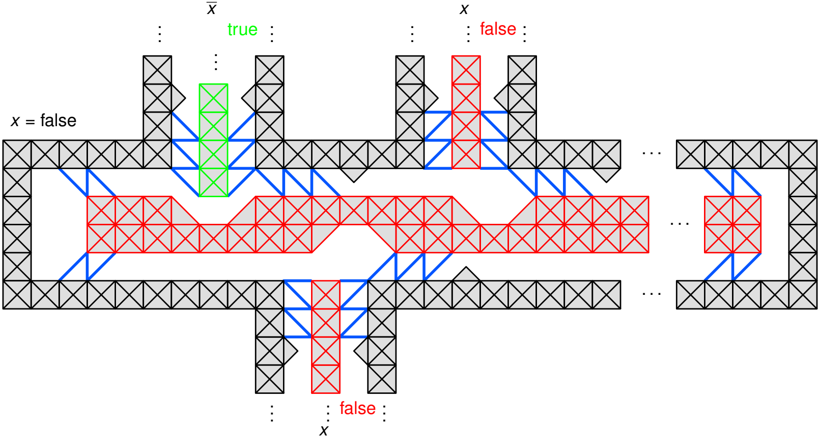
Input: planar 3-SAT formula $\varphi = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \ldots$

Goal: planar embedded graph G_φ with:
G_φ has a metro map drawing $\Leftrightarrow$ φ satisfiable.

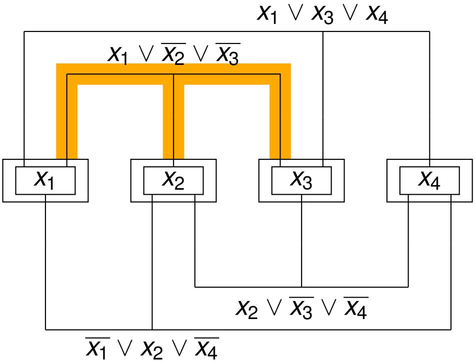
Variable Gadget



Variable Gadget



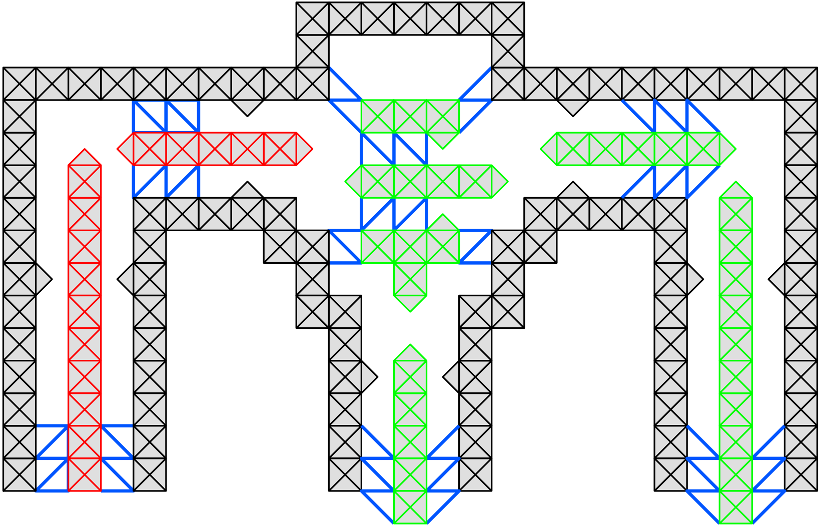
Outline of the Reduction



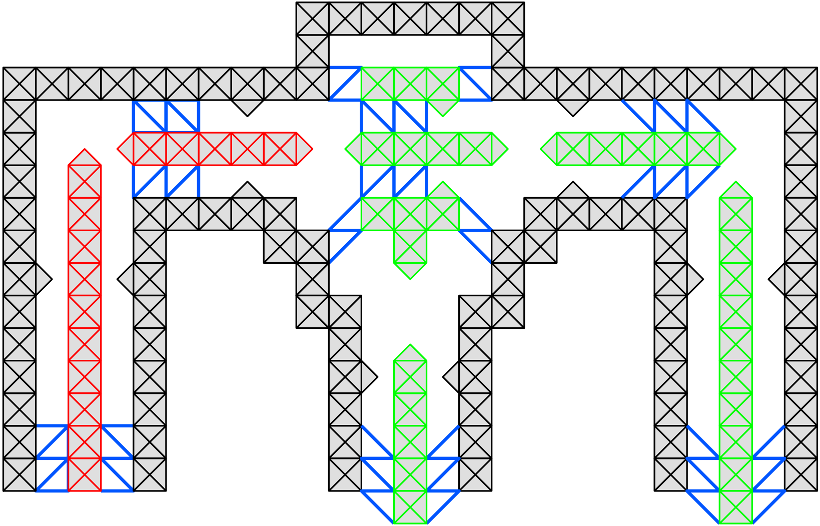
Input: planar 3-SAT formula $\varphi = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \ldots$

Goal: planar embedded graph G_φ with:
G_φ has a metro map drawing $\Leftrightarrow$ φ satisfiable.

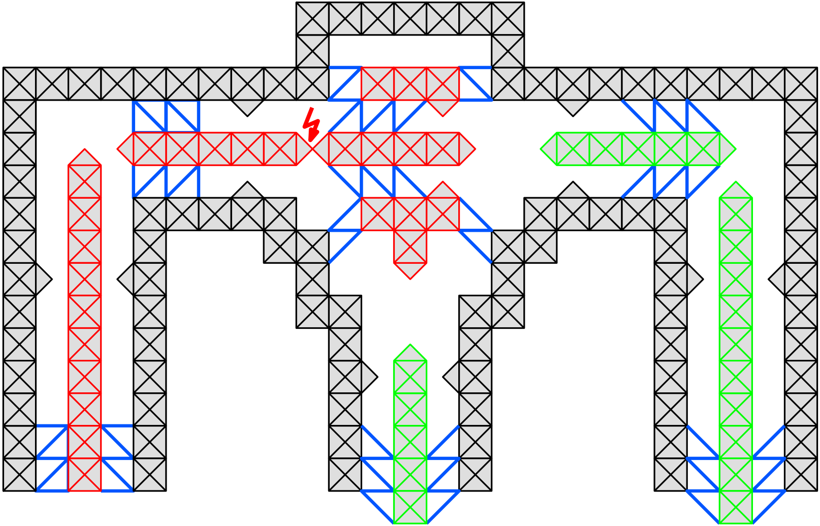
Clause Gadget



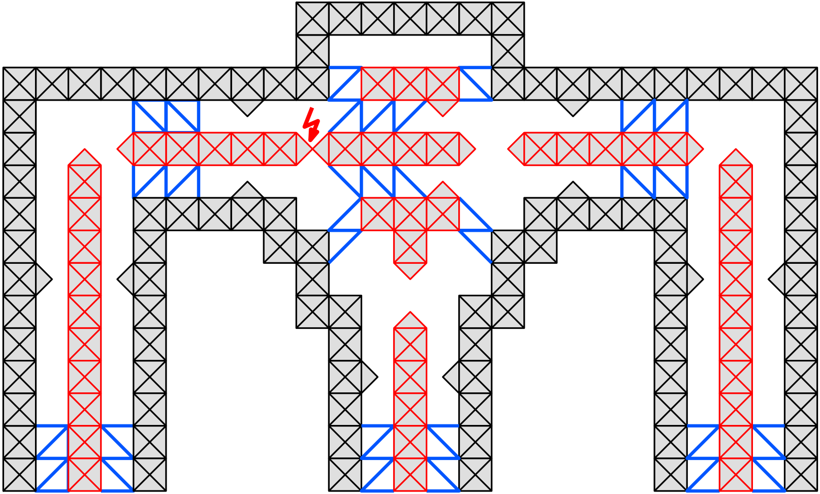
Clause Gadget



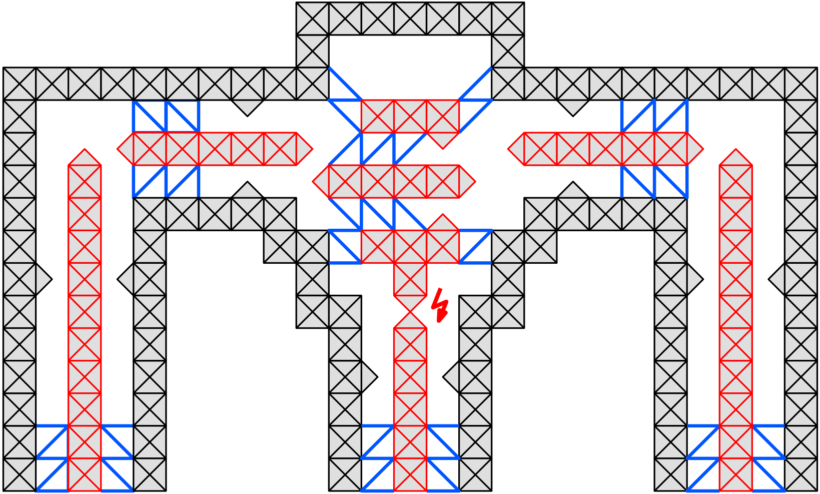
Clause Gadget



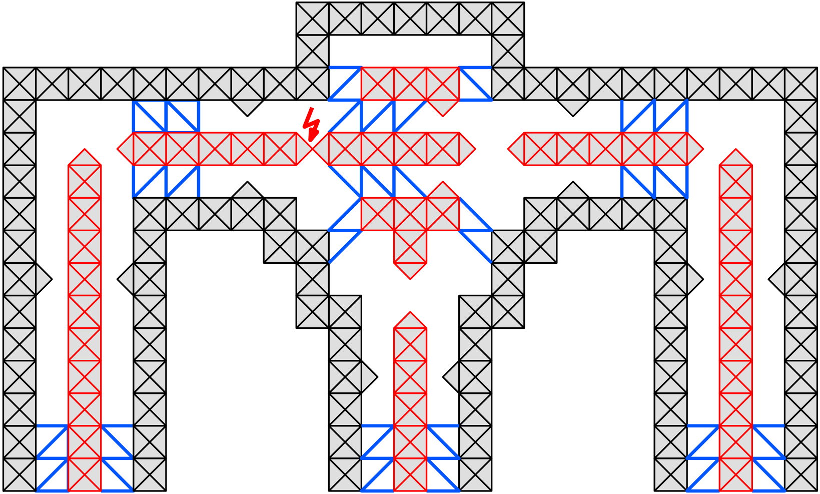
Clause Gadget



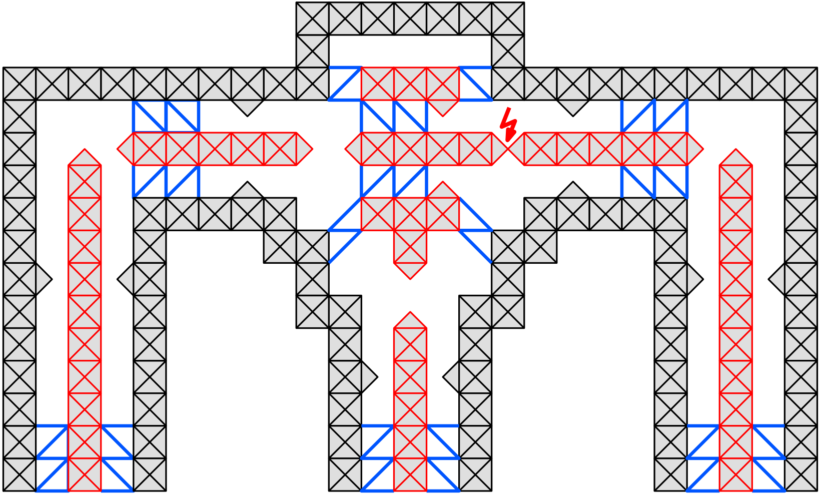
Clause Gadget



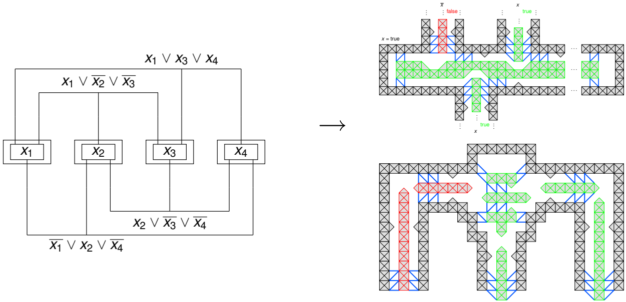
Clause Gadget



Clause Gadget



Summary of the Reduction



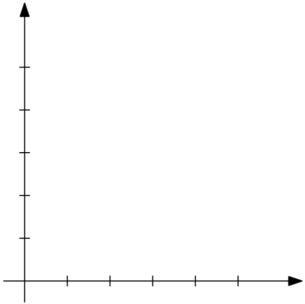
- Indeed we have:
  - φ satisfiable $\Rightarrow$ corresponding MM drawing of G_φ
  - G_φ has MM drawing $\Rightarrow$ satisfying truth assignment of φ

Outline

- Modeling the Metro Map Problem
  - What is a Metro Map?
  - Hard and Soft Constraints
- NP-Hardness: Bad News—Nice Proof
  - Rectilinear vs. Octilinear Drawing
  - Reduction from PLANAR 3-SAT
- MIP Formulation & Experiments
  - Mixed-Integer Programming Formulation
  - Experiments
  - Labeling

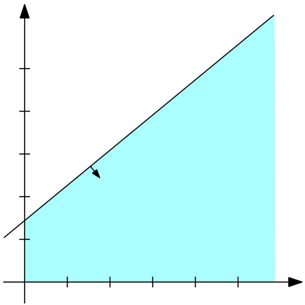
Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables



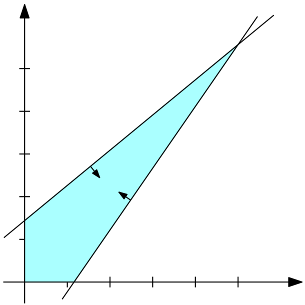
Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
  - example:
    maximize $x + 2y$
    subject to
    $y \leq 0.9x + 1.5$
    $y \geq 1.4x - 1.3$



Mathematical Programming

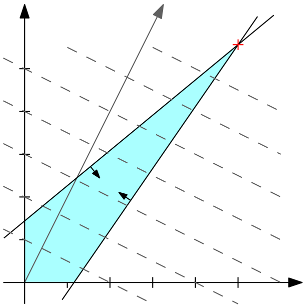
- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
  - example:
    maximize $x + 2y$
    subject to
    $y \leq 0.9x + 1.5$
    $y \geq 1.4x - 1.3$



Mathematical Programming

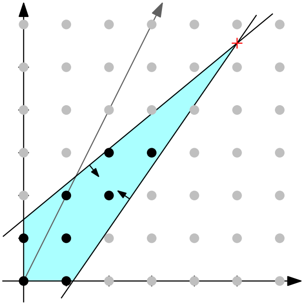
- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
  - example:

    maximize $x + 2y$
    subject to
    $y \leq 0.9x + 1.5$
    $y \geq 1.4x - 1.3$



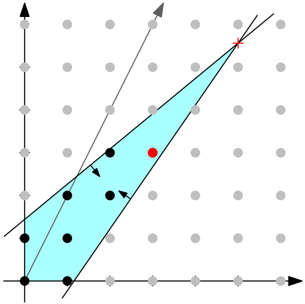
Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
- Mixed-Integer Programming (MIP)
  - allows also integer variables
  - NP-hard in general
  - still practical method for many hard optimization problems



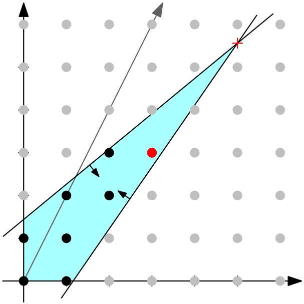
Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
- Mixed-Integer Programming (MIP)
  - allows also integer variables
  - NP-hard in general
  - still practical method for many hard optimization problems



Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
- Mixed-Integer Programming (MIP)
  - allows also integer variables
  - NP-hard in general
  - still practical method for many hard optimization problems

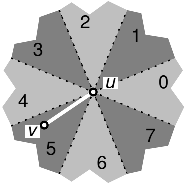


Theorem (Nöllenburg & Wolff GD'05)

The problem MetroMapLayout can be formulated as a MIP s.th.

hard constraints → linear constraints
soft constraints → objective function

Definitions: Sectors and Coordinates

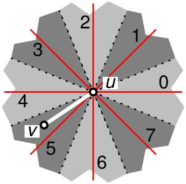


Sectors

– for each vtx. u partition plane into sectors 0–7

- here: $\mathrm{sec}(u, v) = 5$ (input)

Definitions: Sectors and Coordinates



Sectors

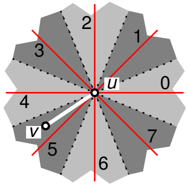
– for each vtx. u partition plane into sectors 0–7

- here: $\sec(u, v) = 5$ (input)

– number octilinear edge directions accordingly

- e.g. $\operatorname{dir}(u, v) = 4$ (output)

Definitions: Sectors and Coordinates



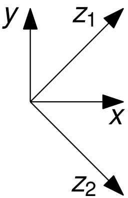
Sectors

– for each vtx. u partition plane into sectors 0–7

- here: $\sec(u, v) = 5$ (input)

– number octilinear edge directions accordingly

- e.g. $\text{dir}(u, v) = 4$ (output)



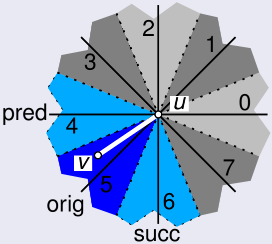
Coordinates

assign z_1- and z_2-coordinates to each vertex v:

- $z_1(v) = x(v) + y(v)$
- $z_2(v) = x(v) - y(v)$

Octilinearity and Relative Position

Goal

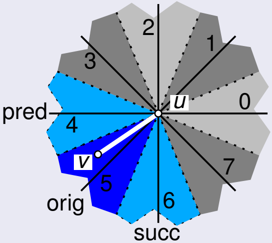


Draw edge *uv*

- octilinearly
- with minimum length ℓ_{uv}
- restricted to 3 directions

Octilinearity and Relative Position

Goal



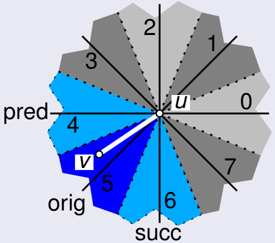
Draw edge *uv*

- octilinearly
- with minimum length ℓ_{uv}
- restricted to 3 directions

How to model this using linear constraints?

Octilinearity and Relative Position

Goal



Draw edge *uv*

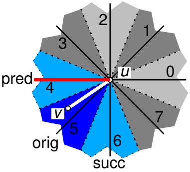
- octilinearly
- with minimum length ℓ_{uv}
- restricted to 3 directions

How to model this using linear constraints?

Binary Variables

$$\alpha_{\text{pred}}(u, v) + \alpha_{\text{orig}}(u, v) + \alpha_{\text{succ}}(u, v) = 1$$

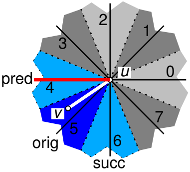
Octilinearity and Relative Position



Predecessor Sector

$$
\begin{aligned}
y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\
-y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\
x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u, v)) + \ell_{uv}
\end{aligned}
$$

Octilinearity and Relative Position

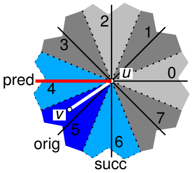


Predecessor Sector

$$
\begin{aligned}
y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\
-y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\
x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u, v)) + \ell_{uv}
\end{aligned}
$$

How does this work?

Octilinearity and Relative Position



Predecessor Sector

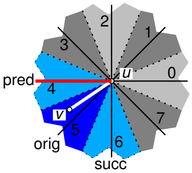
$$
\begin{aligned}
y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\
-y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\
x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u, v)) + \ell_{uv}
\end{aligned}
$$

How does this work?

Case 1: $\alpha_{\text{pred}}(u, v) = 0$

$$
\begin{aligned}
y(u) - y(v) &\leq M \\
-y(u) + y(v) &\leq M \\
x(u) - x(v) &\geq \ell_{uv} - M
\end{aligned}
$$

Octilinearity and Relative Position



Predecessor Sector

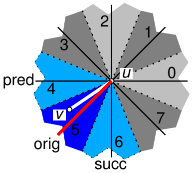
$$\begin{aligned} y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ -y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u,v)) + \ell_{uv} \end{aligned}$$

How does this work?

Case 2: $\alpha_{\text{prev}}(u,v) = 1$

$$\begin{aligned} y(u) - y(v) &\leq 0 \\ -y(u) + y(v) &\leq 0 \\ x(u) - x(v) &\geq \ell_{uv} \end{aligned}$$

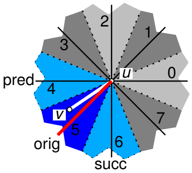
Octilinearity and Relative Position



Original Sector

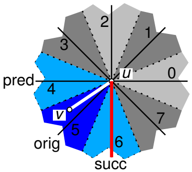
$$\begin{aligned} z_2(u) - z_2(v) &\leq M(1 - \alpha_{\text{orig}}(u,v)) \\ -z_2(u) + z_2(v) &\leq M(1 - \alpha_{\text{orig}}(u,v)) \\ z_1(u) - z_1(v) &\geq -M(1 - \alpha_{\text{orig}}(u,v)) + 2\ell_{uv} \end{aligned}$$

Octilinearity and Relative Position



Original Sector

$$
\begin{aligned}
z_2(u) - z_2(v) &\leq M(1 - \alpha_{\mathrm{orig}}(u, v)) \\
-z_2(u) + z_2(v) &\leq M(1 - \alpha_{\mathrm{orig}}(u, v)) \\
z_1(u) - z_1(v) &\geq -M(1 - \alpha_{\mathrm{orig}}(u, v)) + 2\ell_{uv}
\end{aligned}
$$



Successor Sector

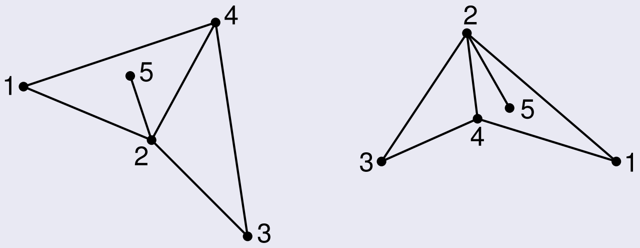
$$
\begin{aligned}
x(u) - x(v) &\leq M(1 - \alpha_{\mathrm{succ}}(u, v)) \\
-x(u) + x(v) &\leq M(1 - \alpha_{\mathrm{succ}}(u, v)) \\
y(u) - y(v) &\geq -M(1 - \alpha_{\mathrm{succ}}(u, v)) + \ell_{uv}
\end{aligned}
$$

Preserving the Embedding

Definition

Two planar drawings of G have the same *embedding* if the induced orderings on the neighbors of each vertex are equal.

Same Embedding

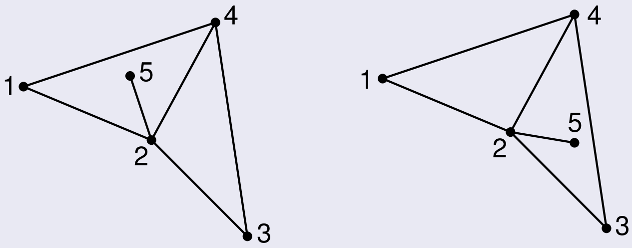


Preserving the Embedding

Definition

Two planar drawings of G have the same *embedding* if the induced orderings on the neighbors of each vertex are equal.

Different Embeddings



Preserving the Embedding

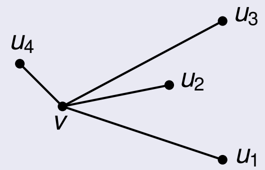
Constraints (Example)

- $N(v) = \{u_1, u_2, u_3, u_4\}$
- circular input order: $u_1 < u_2 < u_3 < u_4 < u_1$

All but one of the following inequalities must hold

$$\mathrm{dir}(v, u_1) < \mathrm{dir}(v, u_2) < \mathrm{dir}(v, u_3) < \mathrm{dir}(v, u_4) < \mathrm{dir}(v, u_1)$$

Input



Preserving the Embedding

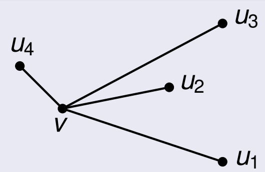
Constraints (Example)

- $N(v) = \{u_1, u_2, u_3, u_4\}$
- circular input order: $u_1 < u_2 < u_3 < u_4 < u_1$

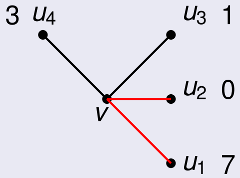
All but one of the following inequalities must hold

$$\text{dir}(v, u_1) \not< \text{dir}(v, u_2) < \text{dir}(v, u_3) < \text{dir}(v, u_4) < \text{dir}(v, u_1)$$

Input



Output

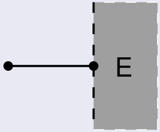


Planarity

Observation

For octilinear, straight edge e_1
non-intersecting edge e_2 must be placed

- east, northeast, north, northwest, west, southwest, south, or southeast

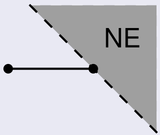


Planarity

Observation

For octilinear, straight edge e_1
non-intersecting edge e_2 must be placed

- east, northeast, north, northwest, west, southwest, south, or southeast



Planarity

Observation

For octilinear, straight edge e_1
non-intersecting edge e_2 must be placed

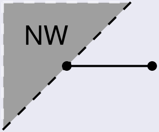
- east, northeast, north, northwest, west, southwest, south, or southeast

Planarity

Observation

For octilinear, straight edge e_1
non-intersecting edge e_2 must be placed

- east, northeast, north, northwest, west, southwest, south, or southeast

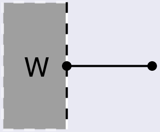


Planarity

Observation

For octilinear, straight edge e_1
non-intersecting edge e_2 must be placed

- east, northeast, north, northwest, west, southwest, south, or southeast

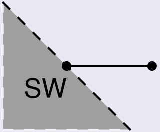


Planarity

Observation

For octilinear, straight edge e_1
non-intersecting edge e_2 must be placed

- east, northeast, north, northwest, west, southwest, south, or southeast

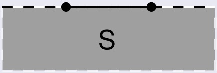


Planarity

Observation

For octilinear, straight edge e_1
non-intersecting edge e_2 must be placed

- east, northeast, north, northwest, west, southwest, south, or southeast

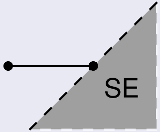


Planarity

Observation

For octilinear, straight edge e_1
non-intersecting edge e_2 must be placed

- east, northeast, north, northwest, west, southwest, south, or southeast

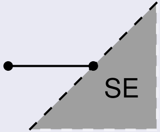


Planarity

Observation

For octilinear, straight edge e_1
non-intersecting edge e_2 must be placed

- east, northeast, north, northwest, west, southwest, south, or southeast



Constraints

- model as MIP with binary variables
- need planarity constraints for each pair of non-incident edges

Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\ \text{cost}_{\text{bends}} + \lambda_{\text{length}}\ \text{cost}_{\text{length}} + \lambda_{\text{relpos}}\ \text{cost}_{\text{relpos}}$

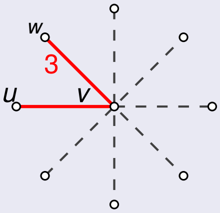
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges *uv* and *vw* on a line $L \in \mathcal{L}$

- draw as straight as possible
- increasing cost $bend(u, v, w)$ for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

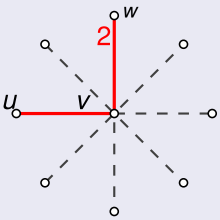
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a line $L \in \mathcal{L}$

- draw as straight as possible
- increasing cost $bend(u, v, w)$ for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

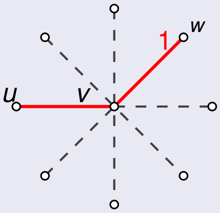
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a line $L \in \mathcal{L}$

- draw as straight as possible
- increasing cost $bend(u, v, w)$ for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

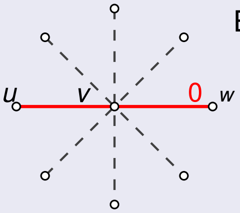
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a line $L \in \mathcal{L}$

- draw as straight as possible
- increasing cost $bend(u, v, w)$ for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

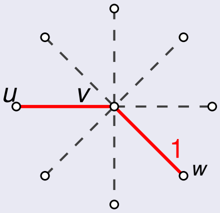
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a line $L \in \mathcal{L}$

- draw as straight as possible
- increasing cost $bend(u, v, w)$ for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

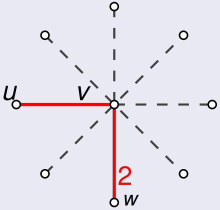
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\, \text{cost}_{\text{bends}} + \lambda_{\text{length}}\, \text{cost}_{\text{length}} + \lambda_{\text{relpos}}\, \text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a line $L \in \mathcal{L}$

- draw as straight as possible
- increasing cost $bend(u, v, w)$ for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

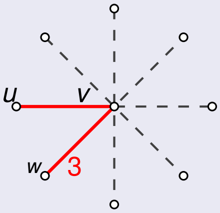
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a line $L \in \mathcal{L}$

- draw as straight as possible
- increasing cost $bend(u, v, w)$ for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

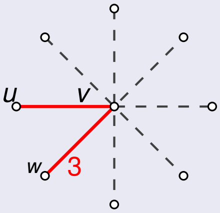
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\, \text{cost}_{\text{bends}} + \lambda_{\text{length}}\, \text{cost}_{\text{length}} + \lambda_{\text{relpos}}\, \text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a line $L \in \mathcal{L}$

- draw as straight as possible
- increasing cost $bend(u, v, w)$ for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

$$\text{cost}_{\text{bends}} = \sum_{L \in \mathcal{L}} \sum_{uv, vw \in L} \text{bend}(u, v, w)$$

Objective Function

Total Edge Length (S2)

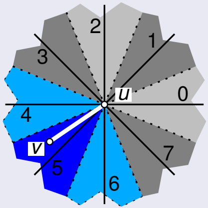
$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Objective Function

Total Edge Length (S2)

$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Relative Position (S3)



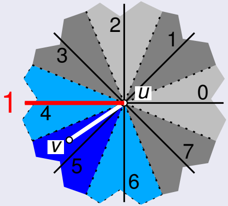
- only three directions possible

Objective Function

Total Edge Length (S2)

$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Relative Position (S3)



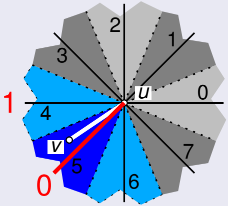
- only three directions possible
- charge 1 if edge deviates from original sector

Objective Function

Total Edge Length (S2)

$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Relative Position (S3)



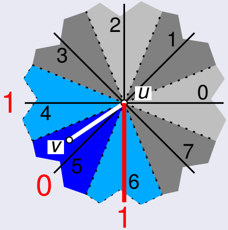
- only three directions possible
- charge 1 if edge deviates from original sector

Objective Function

Total Edge Length (S2)

$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Relative Position (S3)



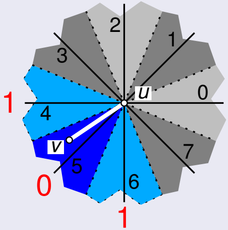
- only three directions possible
- charge 1 if edge deviates from original sector

Objective Function

Total Edge Length (S2)

$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Relative Position (S3)



- only three directions possible
- charge 1 if edge deviates from original sector

$$\text{cost}_{\text{relpos}} = \sum_{uv \in E} \text{relpos}(uv)$$

Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity

Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity
- soft constraints:

  minimize $\lambda_{\text{bends}} \text{cost}_{\text{bends}} + \lambda_{\text{length}} \text{cost}_{\text{length}} + \lambda_{\text{relpos}} \text{cost}_{\text{relpos}}$

Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity
- soft constraints:

  minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$
- models METROMAPLAYOUT as MIP

Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity
- soft constraints:

  minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

- models METROMAPLAYOUT as MIP
- in total $O(|V|^2)$ constraints and variables

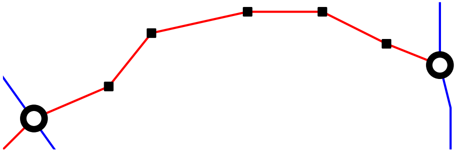
Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity
- soft constraints:

  minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

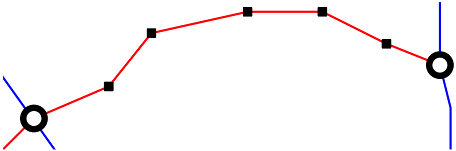
- models METROMAPLAYOUT as MIP
- in total $O(|V|^2)$ constraints and variables

Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

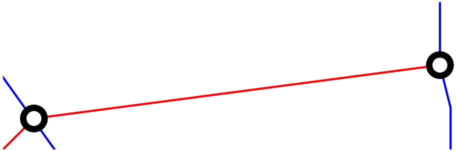
Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

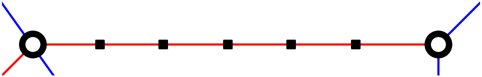
Idea 1 collapse all degree-2 vertices

Speed-Up Techniques: Reduce Graph Size

- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

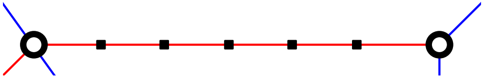
Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

Speed-Up Techniques: Reduce Graph Size

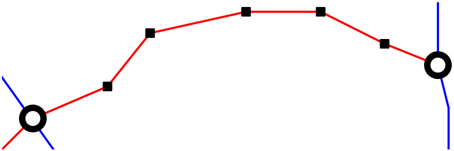


- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

- low flexibility

Speed-Up Techniques: Reduce Graph Size

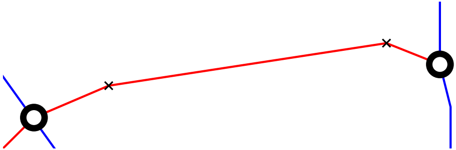


- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices
- low flexibility

Idea 2 keep two *joints*

Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices
- low flexibility

Idea 2 keep two *joints*

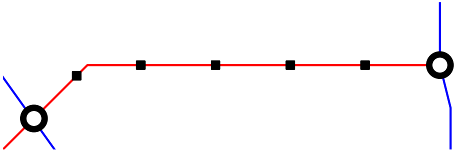
Speed-Up Techniques: Reduce Graph Size

- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices
- low flexibility

Idea 2 keep two *joints*

Speed-Up Techniques: Reduce Graph Size



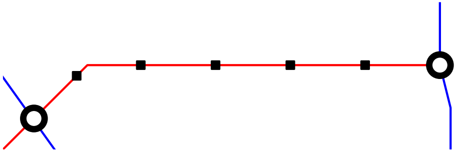
- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

- low flexibility

Idea 2 keep two *joints*

Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices
- low flexibility

Idea 2 keep two *joints*
- higher flexibility
- more similar to input

Speed-Up Techniques: Reduce MIP Size

- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

Speed-Up Techniques: Reduce MIP Size

- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

Observation 1

- consider only pairs of edges incident to the same face
- still $O(|V|^2)$ constraints

Speed-Up Techniques: Reduce MIP Size

- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

Observation 1

- consider only pairs of edges incident to the same face
- still $O(|V|^2)$ constraints

Observation 2

- in practice no or only few crossings due to soft constraints

Speed-Up Techniques: Reduce MIP Size

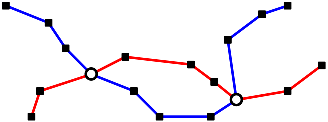
- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

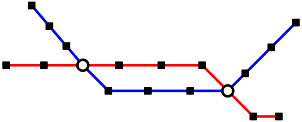
Observation 1

- consider only pairs of edges incident to the same face
- still $O(|V|^2)$ constraints

Observation 2

- in practice no or only few crossings due to soft constraints





Speed-Up Techniques: Reduce MIP Size

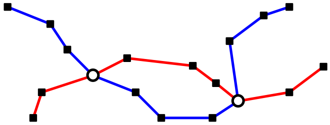
- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

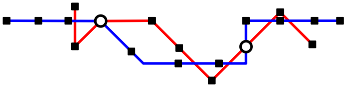
Observation 1

- consider only pairs of edges incident to the same face
- still $O(|V|^2)$ constraints

Observation 2

- in practice no or only few crossings due to soft constraints





Speed-Up Techniques: Callback Functions

- MIP optimizer CPLEX offers advanced callback functions
- add required planarity constraints on the fly

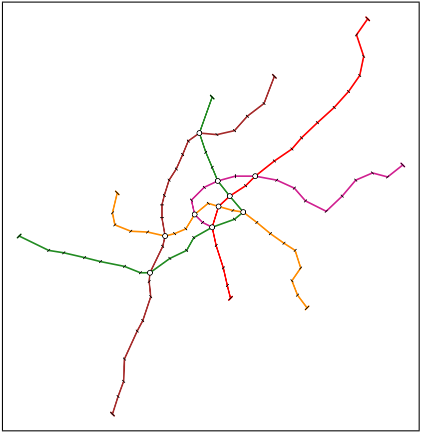
Algorithm

1. start solving MIP without planarity constraints
2. for each new solution s
    1. interrupt CPLEX
    2. if s is not planar
        - add planarity constraints for edges that intersect in s
        - reject s

       else
        - accept s
    3. continue solving the MIP (until optimal)

Results – Vienna

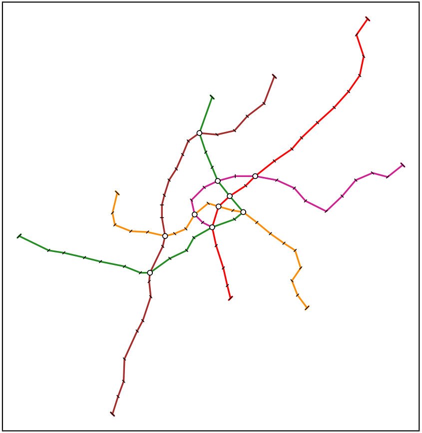
Input



Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
normal	90	96	8	5
reduced	44	50		

Results – Vienna

Input



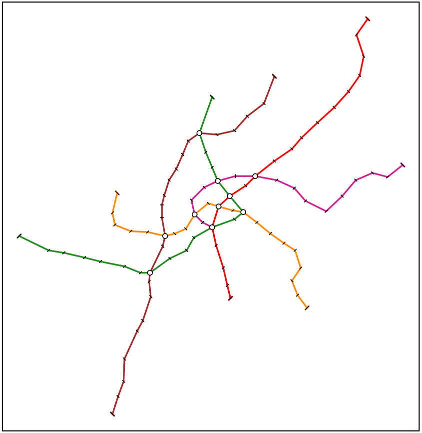
Input	$\lvert V \rvert$	$\lvert E \rvert$	fcs.	$\lvert \mathcal{L} \rvert$
normal	90	96	8	5
reduced	44	50		

$\downarrow$

MIP	constr.	var.
normal	39,363	9,960
faces	23,226	6,048
callback	1,875	872

Results – Vienna

Input



Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
normal	90	96	8	5
reduced	44	50		

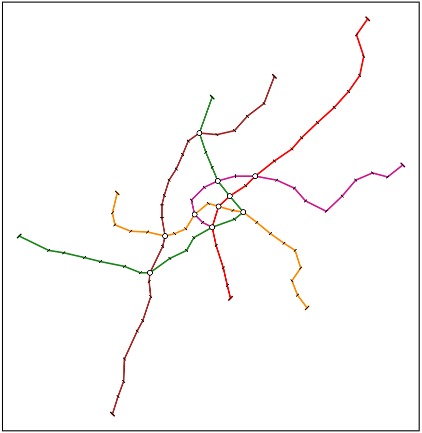
↓

MIP	constr.	var.
normal	39,363	9,960
faces	23,226	6,048
callback*	1,875	872

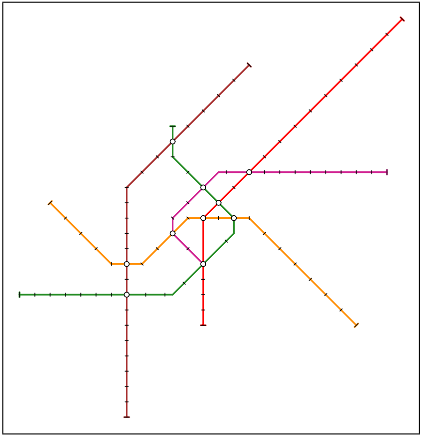
*) 21 seconds w/o proof of opt.

Results – Vienna

Input

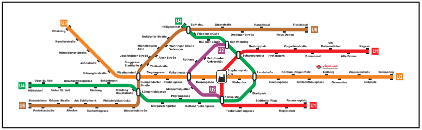


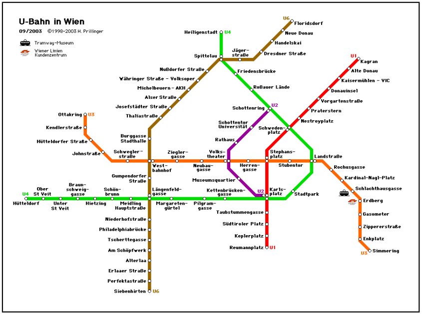
Output (21 sec.)



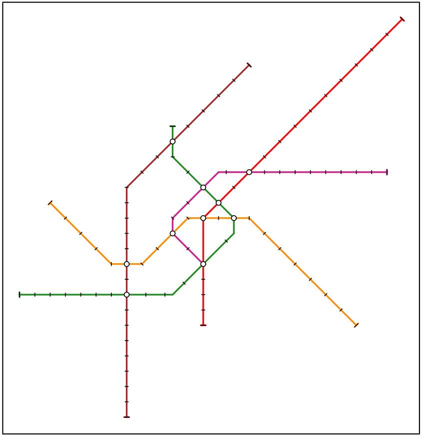
Results – Vienna

Official maps



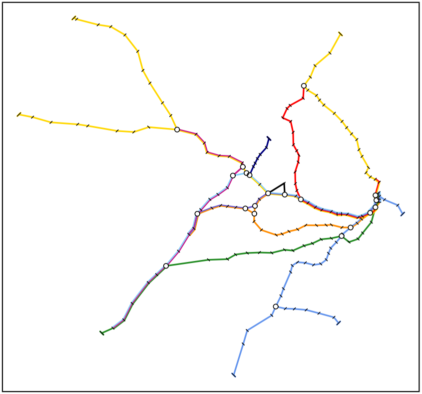


Output (21 sec.)



Results – Sydney

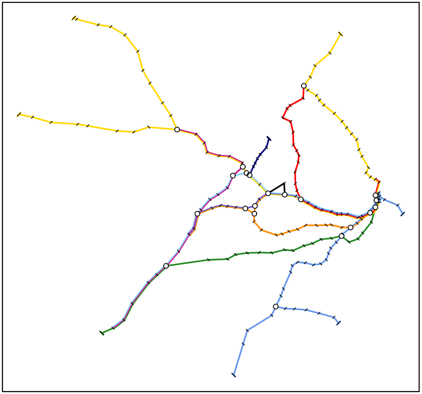
Input



Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
normal	174	183	11	10
reduced	67	76		

Results – Sydney

Input



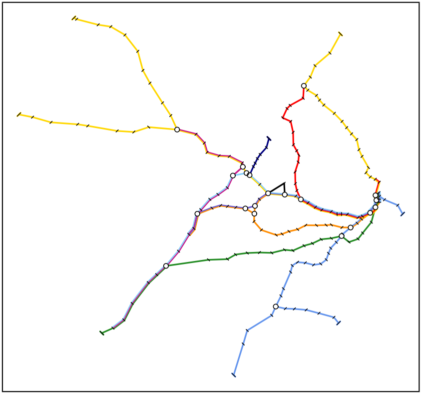
Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
normal	174	183	11	10
reduced	67	76		

↓

MIP	constr.	var.
normal	93,620	23,389
faces	52,568	13,437
callback	4,147	6,741

Results – Sydney

Input



Input	$\lvert V\rvert$	$\lvert E\rvert$	fcs.	$\lvert\mathcal{L}\rvert$
normal	174	183	11	10
reduced	67	76		

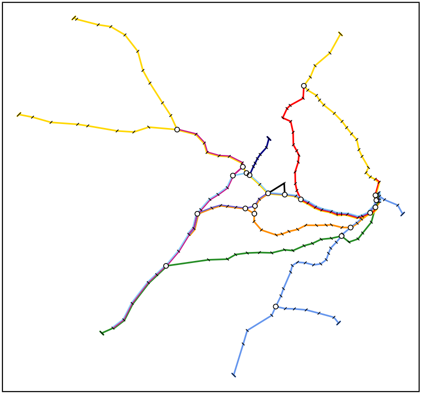
↓

MIP	constr.	var.
normal	93,620	23,389
faces	52,568	13,437
callback*	4,147	6,741

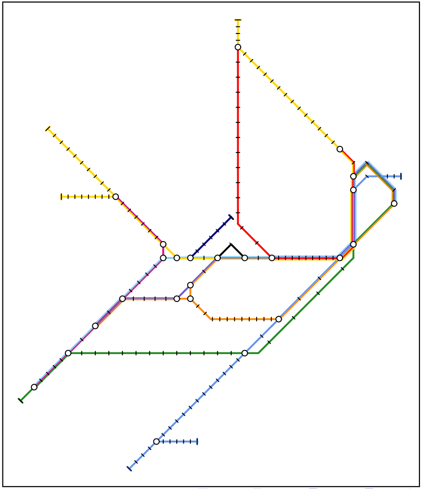
*) 33 seconds w/o proof of opt.
constr. of 4 edge pairs added

Results – Sydney

Input

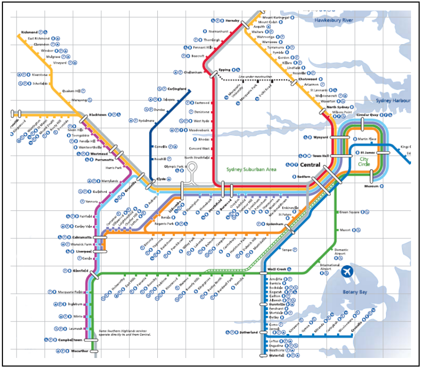


Output (33 sec.)

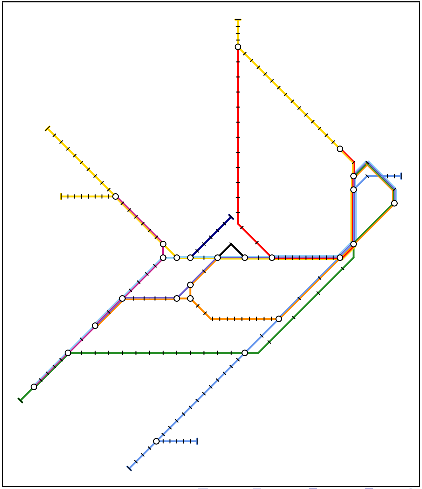


Results – Sydney

Official map

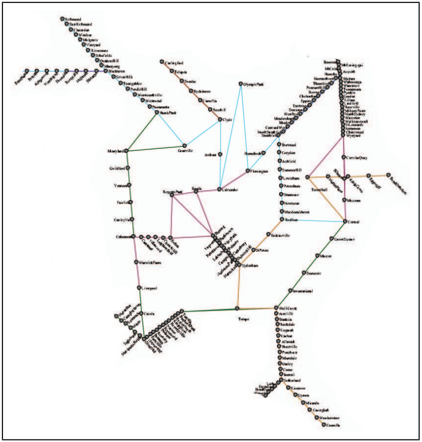


Output (33 sec.)

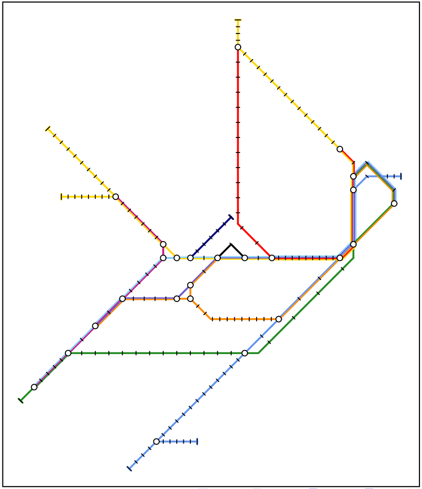


Sydney: Related Work

[Hong et al. GD'04] (7.6 sec.)

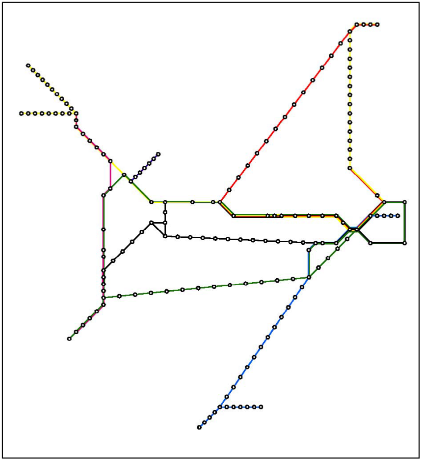


Our output (33 sec.)

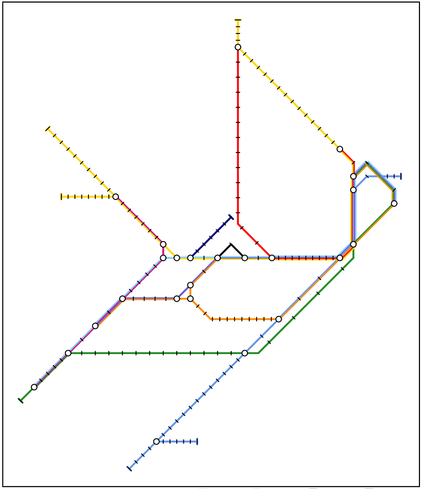


Sydney: Related Work

[Stott, Rodgers IV'04] (4 min.)

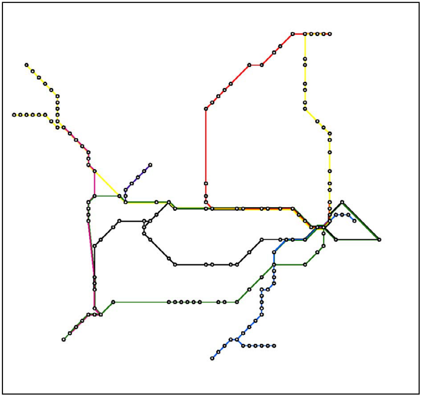


Our output (33 sec.)

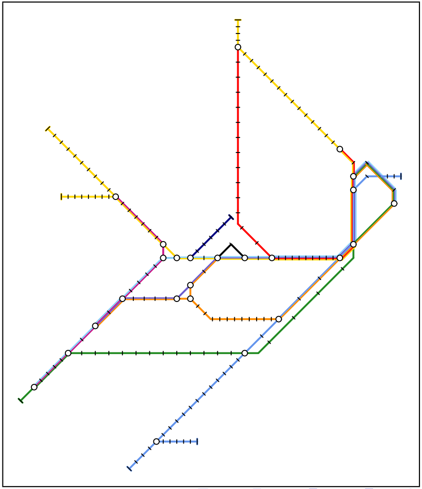


Sydney: Related Work

[Stott, Rodgers IV'04] (28 min.)

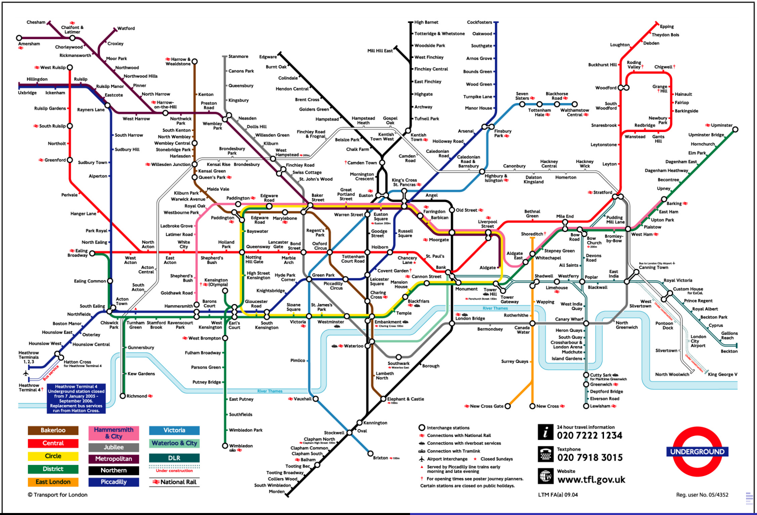


Our output (33 sec.)



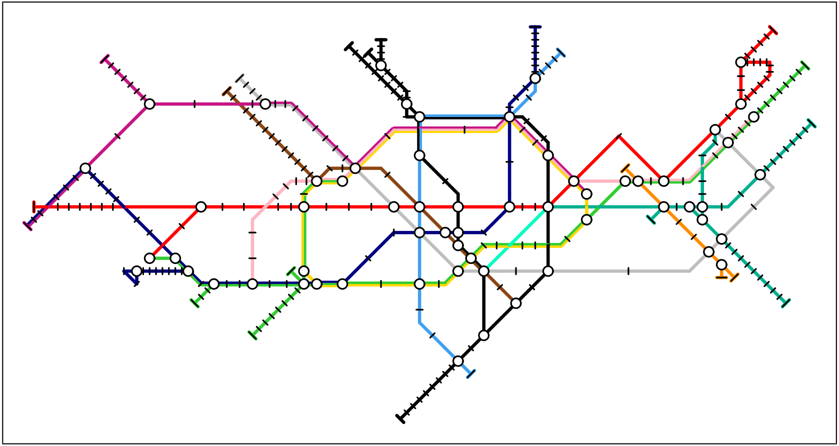
Large Maps

London



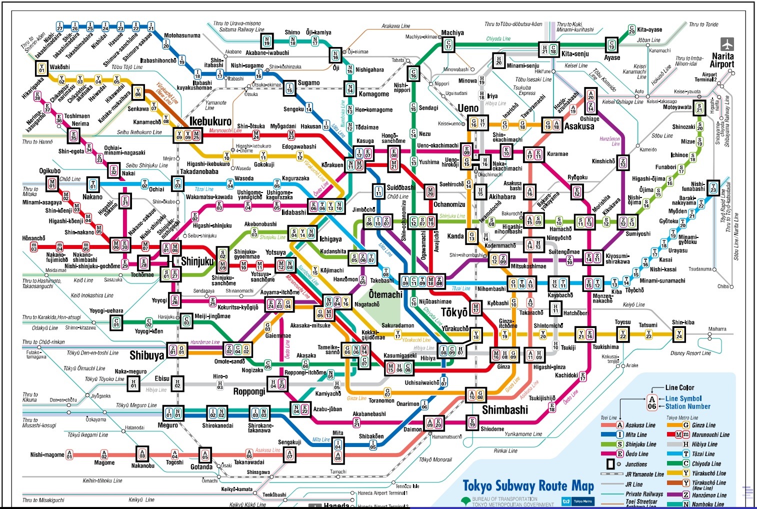
Large Maps

London (16 min.)



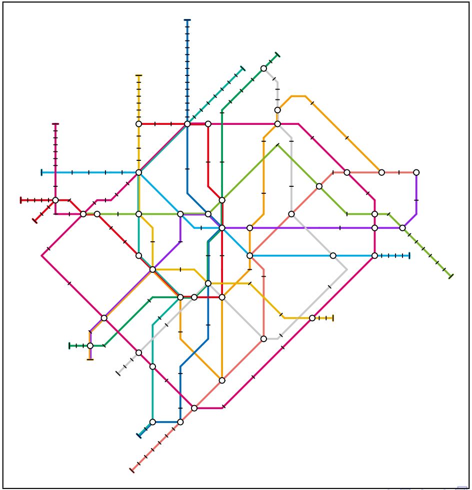
Large Maps

Tokyo



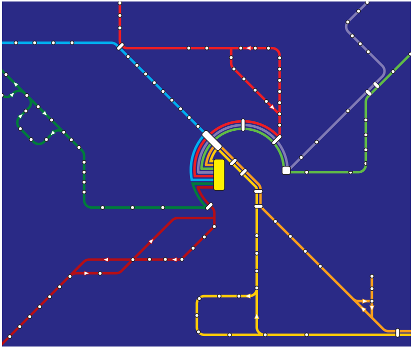
Large Maps

Tokyo (4:50 hrs.)



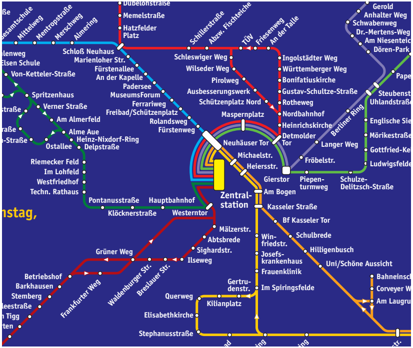
Labeling

- unlabeled metro map of little use in practice



Labeling

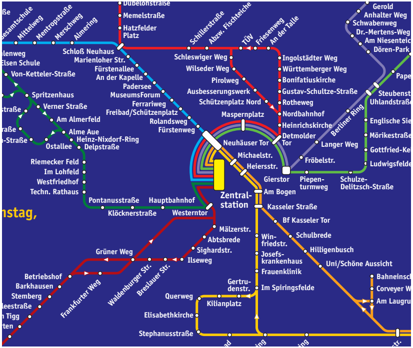
- unlabeled metro map of little use in practice
- labels
  - occupy space
  - may not overlap



Labeling

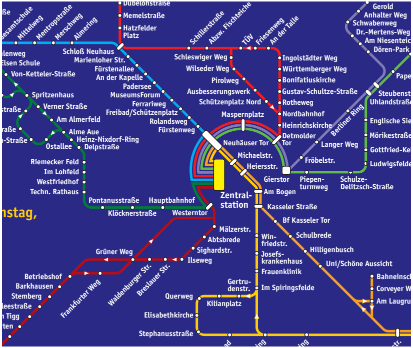
- unlabeled metro map of little use in practice
- labels
  - occupy space
  - may not overlap
- static map labeling is NP-hard

  [Tollis, Kakoulis ’01]



Labeling

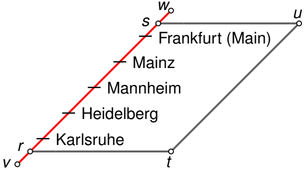
- unlabeled metro map of little use in practice
- labels
  - occupy space
  - may not overlap
- static map labeling is NP-hard
  [Tollis, Kakoulis '01]
- want to combine layout and labeling for better results



Modeling Labels

Model labels as special metro lines:

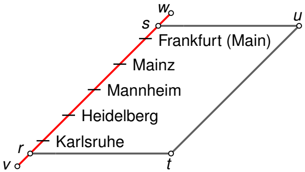
- put all labels between a pair of interchange stations into one parallelogram,



Modeling Labels

Model labels as special metro lines:

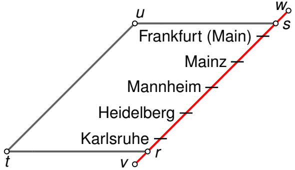
- put all labels between a pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,



Modeling Labels

Model labels as special metro lines:

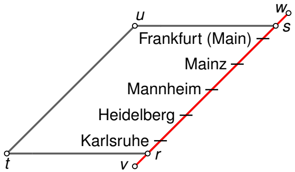
- put all labels between a pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,



Modeling Labels

Model labels as special metro lines:

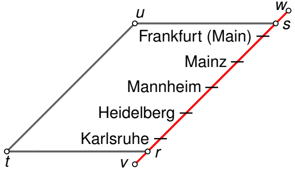
- put all labels between a pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,
- bad news: a **lot** more planarity constraints



Modeling Labels

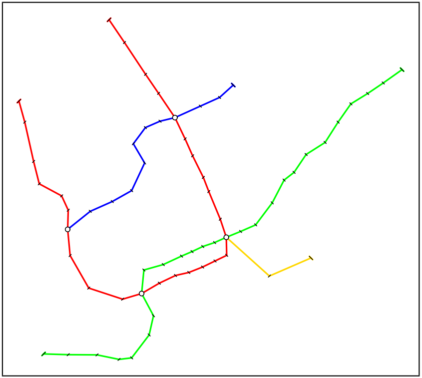
Model labels as special metro lines:

- put all labels between a pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,
- bad news: a **lot** more planarity constraints
- good news: callback method helps



Labeling – Montréal

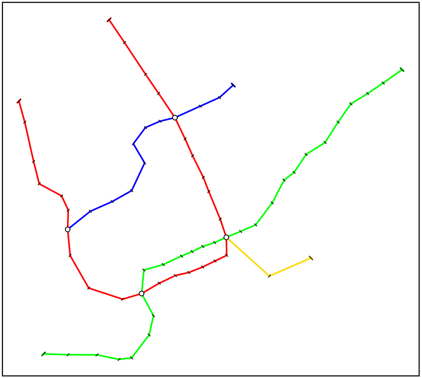
Input



Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
normal	65	66	3	4
reduced	20	21		
labeled	62	74	14	

Labeling – Montréal

Input



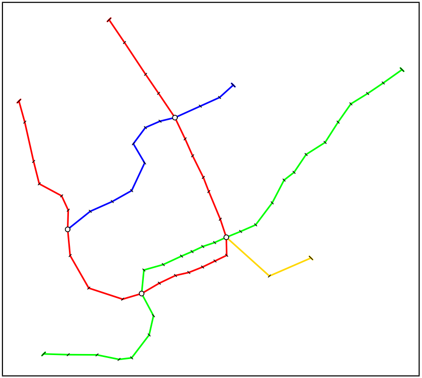
Input	$\lvert V\rvert$	$\lvert E\rvert$	fcs.	$\lvert\mathcal{L}\rvert$
normal	65	66	3	4
reduced	20	21		
labeled	62	74	14	

↓

MIP	constr.	var.
normal	86,493	21,209
faces	32,208	8,049
callback	4,400	8,049

Labeling – Montréal

Input



Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
normal	65	66	3	4
reduced	20	21		
labeled	62	74	14	

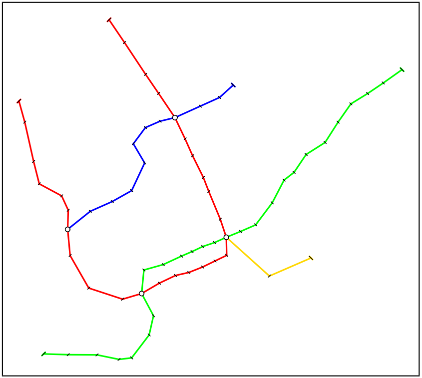
↓

MIP	constr.	var.
normal	86,493	21,209
faces	32,208	8,049
callback*	4,400	8,049

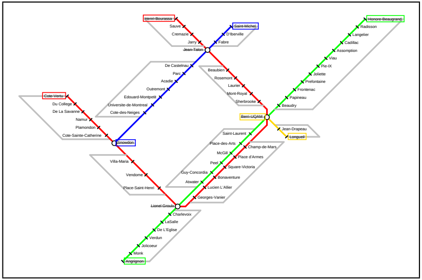
*) 17 minutes w/o proof of opt.
constr. of 60 edge pairs added

Labeling – Montréal

Input



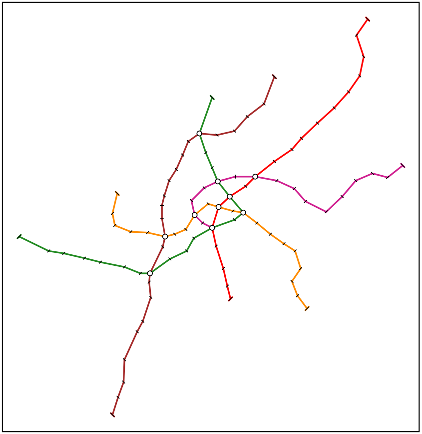
Output (17 min.)



Labeling – Montréal

Labeling – Vienna

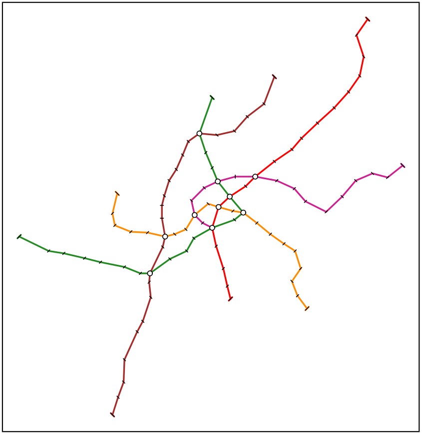
Input



<table>
<tr><th>Input</th><th>$|V|$</th><th>$|E|$</th><th>fcs.</th><th>$|\mathcal{L}|$</th></tr>
<tr><td>normal</td><td>90</td><td>96</td><td rowspan="2">8</td><td rowspan="3">5</td></tr>
<tr><td>reduced</td><td>44</td><td>50</td></tr>
<tr><td>labeled</td><td>98</td><td>117</td><td>21</td></tr>
</table>

Labeling – Vienna

Input



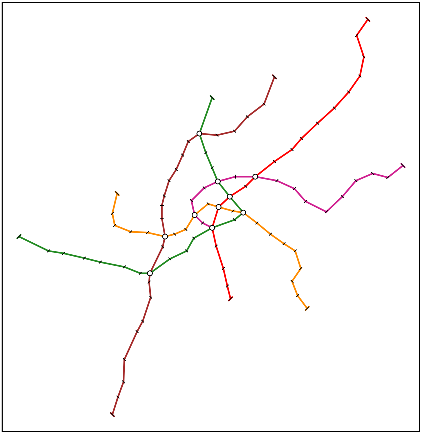
Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
normal	90	96	8	5
reduced	44	50		
labeled	98	117	21	

↓

MIP	constr.	var.
normal	219,064	53,538
faces	51,160	12,834
callback	9,144	12,834

Labeling – Vienna

Input



Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
normal	90	96	8	
reduced	44	50		5
labeled	98	117	21	

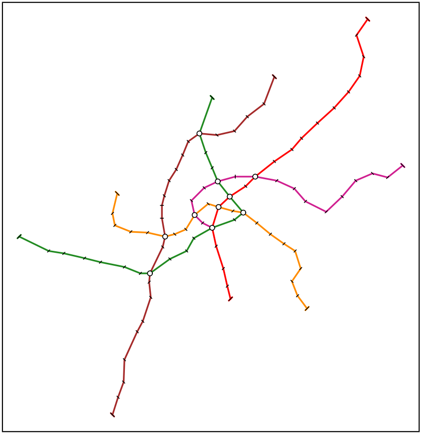
$\downarrow$

MIP	constr.	var.
normal	219,064	53,538
faces	51,160	12,834
callback*	9,144	12,834

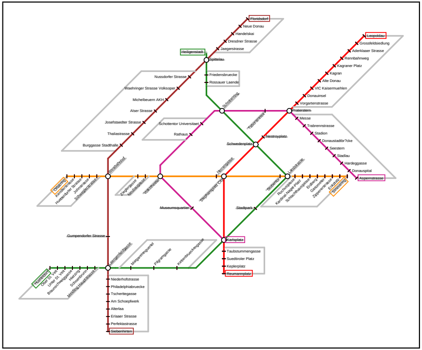
*) 1 day w/o proof of opt.
constr. of 160 edge pairs added

Labeling – Vienna

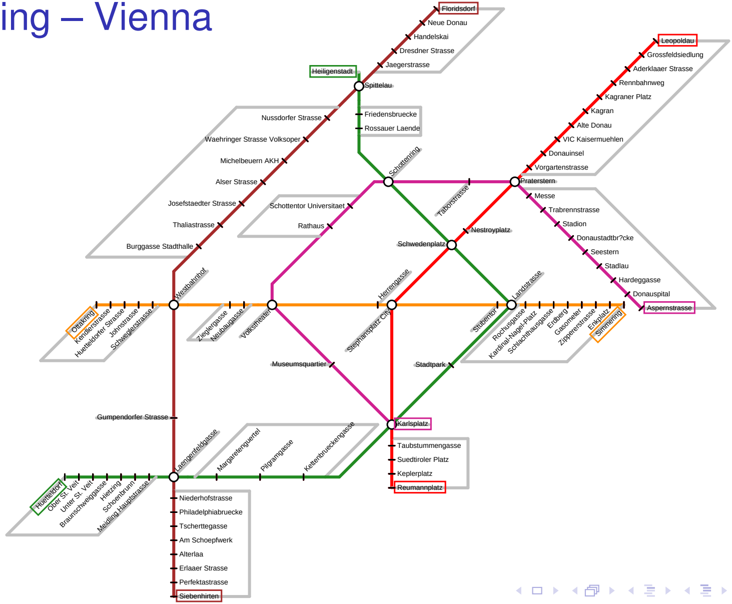
Input



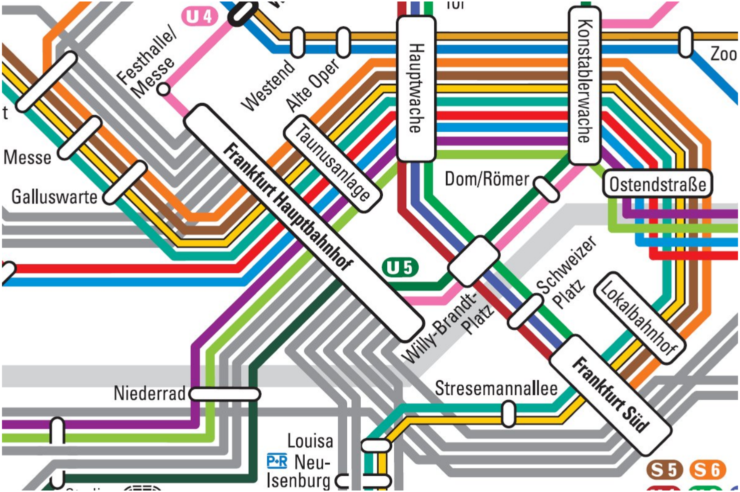
Output (1 day)



Labeling – Vienna



To Do: Rectangular Stations & Multi-Edges



Summary (Metro Maps)

- METROMAPLAYOUT is NP-hard.
- Formulated and implemented MIP.
- Results comparable to manually designed maps.
- Reduced MIP size & runtime drastically.
- Combined layout and labeling.
- Our MIP can draw *any* kind of sketch “nicely”.

Summary (Metro Maps)

- METROMAPLAYOUT is NP-hard.
- Formulated and implemented MIP.
- Results comparable to manually designed maps.
- Reduced MIP size & runtime drastically.
- Combined layout and labeling.
- Our MIP can draw *any* kind of sketch “nicely”.

To Do

- rectangular stations
- multi-edges
- user interaction (e.g. fixing certain edge directions)

Thank you for the attention.

Any questions?